# Table of Contents

Table of Contents .................................................................................................................. 2  
List of Figures ........................................................................................................................ 4  
List of Tables .......................................................................................................................... 6  
1. Introduction of development board .................................................................................. 7  
   1.1. Picture of real development board .............................................................................. 7  
      1.1.1. The EVAL development board ............................................................................ 7  
      1.1.2. The START development board ....................................................................... 8  
   1.2. Boot mode ................................................................................................................... 9  
   1.3. Debugger interface ...................................................................................................... 9  
   1.4. View log .................................................................................................................... 10  
2. Build development environment ...................................................................................... 11  
   2.1. Keil MDK5 installation .............................................................................................. 11  
   2.2. IAR Installation ......................................................................................................... 12  
   2.3. CMAKE+GCC installation ......................................................................................... 12  
      2.3.1. JLINK driver installation .................................................................................... 13  
3. Development instructions ............................................................................................... 14  
   3.1. SDK execution program group ................................................................................... 14  
   3.2. SDK configuration ..................................................................................................... 15  
      3.2.1. Platform configuration ....................................................................................... 15  
      3.2.2. SRAM layout ..................................................................................................... 16  
      3.2.3. FLASH layout .................................................................................................... 16  
      3.2.4. Firmware version number ................................................................................ 17  
      3.2.5. APP configuration ............................................................................................. 17  
   3.3. Firmware generation and download .......................................................................... 17  
   3.4. Example of correct log .............................................................................................. 18  
4. KEIL project ..................................................................................................................... 19  
   4.1. Project group ............................................................................................................. 19  
   4.2. Project configuration ................................................................................................. 20  
      4.2.1. Device selection .................................................................................................. 20  
      4.2.2. Debugger configuration ...................................................................................... 20  
   4.3. Compilation .............................................................................................................. 22  
   4.4. Download .................................................................................................................. 24
# GD32W51x Rapid Development Guide

4.5. Debug ...................................................................................................................... 24

5. IAR project .................................................................................................................. 26
   5.1. Project group ........................................................................................................ 26
   5.2. Project configuration ........................................................................................... 26
      5.2.1. Device selection .......................................................................................... 27
      5.2.2. Debugger configuration .............................................................................. 27
   5.3. Compilation ......................................................................................................... 28
   5.4. Download ............................................................................................................ 31
   5.5. Debug .................................................................................................................. 31

6. GCC project ............................................................................................................... 34
   6.1. Project group ...................................................................................................... 34
   6.2. Compilation ......................................................................................................... 34
   6.3. Debug .................................................................................................................. 35
      6.3.1. Start the GDB server .................................................................................. 35
      6.3.2. Start the GDB Client for debugging ............................................................ 36

7. Q&A ............................................................................................................................ 38
   7.1. DAPLINK disk identification ............................................................................ 38
   7.2. CM33 debugging by KEIL+JLINK .................................................................... 39
   7.3. DAP chip firmware upgrade ............................................................................. 40
   7.4. No image error .................................................................................................... 40
   7.5. Code runs in SRAM ............................................................................................ 40

8. Revision history ............................................................................................................ 42
# List of Figures

Figure 1-1. The picture of the EVAL development board .......................................................... 7  
Figure 1-2. The picture of the START development board ....................................................... 8  
Figure 1-3. Pins of J-LINK ........................................................................................................ 10  
Figure 1-4. The serial port configuration .................................................................................. 10  
Figure 2-1. Install the KEIL GD32W51x PACK .................................................................... 11  
Figure 2-2. Install the IAR GD32W51x Addon .................................................................... 12  
Figure 2-3. JLINK driver download ......................................................................................... 13  
Figure 2-4. JLINK driver installation ..................................................................................... 14  
Figure 3-1. Boot process ......................................................................................................... 14  
Figure 3-2. Platform configuration ......................................................................................... 15  
Figure 3-3. SRAM layout ......................................................................................................... 16  
Figure 3-4. FLASH layout ........................................................................................................ 17  
Figure 3-5 Firmware version .................................................................................................. 17  
Figure 3-6. List of devices and drivers ................................................................................... 18  
Figure 3-7. Project boot Information ...................................................................................... 18  
Figure 4-1. KEIL project Group interface ............................................................................. 19  
Figure 4-2. KEIL “Device” options page .............................................................................. 20  
Figure 4-3. KEIL “Debug” options page ................................................................................ 21  
Figure 4-4. KEIL “JLINK Settings Debug” options page ...................................................... 21  
Figure 4-5. KEIL “Flash Download” options page ............................................................... 22  
Figure 4-6. Current project switch ......................................................................................... 23  
Figure 4-7. Batch compilation ............................................................................................... 23  
Figure 4-8. Batch compilation settings .................................................................................. 24  
Figure 4-9. KEIL download result ......................................................................................... 24  
Figure 4-10. Initialization file for debugging ......................................................................... 25  
Figure 4-11. KEIL debugging interface .................................................................................. 25  
Figure 5-1. IAR project group ................................................................................................ 26  
Figure 5-2. IAR “Target” options page .................................................................................. 27  
Figure 5-3. IAR “Debugger” options page .............................................................................. 28  
Figure 5-4. IAR compilation .................................................................................................. 29  
Figure 5-5. IAR batch compilation settings .......................................................................... 30  
Figure 5-6. IAR Build Actions ............................................................................................... 31  
Figure 5-7. IAR download result ........................................................................................... 31  
Figure 5-8. IAR debugging button ........................................................................................ 32  
Figure 5-9. IAR debugging interface ...................................................................................... 32  
Figure 5-10. IAR multi-image debugging ............................................................................. 33  
Figure 6-1. Create the environment variable HOME ............................................................. 35  
Figure 6-2. Start the JLINK GDB Server .............................................................................. 36  
Figure 6-3. GDB debug window ............................................................................................ 37  
Figure 6-4. Common commands ........................................................................................... 37
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-1</td>
<td>Mbed serial port driver installation</td>
<td>38</td>
</tr>
<tr>
<td>7-2</td>
<td>List of device manager serial ports</td>
<td>38</td>
</tr>
<tr>
<td>7-3</td>
<td>KEIL DEBUG options</td>
<td>39</td>
</tr>
<tr>
<td>7-4</td>
<td>Add tools supported by KEIL</td>
<td>39</td>
</tr>
<tr>
<td>7-5</td>
<td>MAINTENANCE disk</td>
<td>40</td>
</tr>
<tr>
<td>7-6</td>
<td>NSPE MAP file</td>
<td>41</td>
</tr>
</tbody>
</table>
List of Tables

Table 1-1. Boot mode ................................................................................................................. 9
Table 8-1. Revision history ........................................................................................................ 42
1. **Introduction of development board**

1.1. **Picture of real development board**

1.1.1. **The EVAL development board**

The EVAL development board is composed of a baseboard and a module. The module is equipped with GD32W51x WiFi chip and the baseboard provides many peripheral test ports, such as SDIO, I2S, IFRP, TSI and so on. In the figure above, the GD32W51x chip integrates FLASH through SIP (System In a Package). FLASH can also choose the way of external, module size unchanged.

For developers, the main focus may be on the following sections of the development board,
which are indicated in the picture above.

- Boot mode (Boot PIN)
- Power supply port (Power supply)
- Querying log (LOG UART)
- Debugger interface (GDLINK or JLINK)
- Reboot (Reset Button)

Note that the JLINK SWD CK PIN is multiplexed with the BOOT1 PIN. To switch the boot mode, switch the “BOOT1 / SWD Switch” jumper cap to the upper BOOT1 end. If Jlink debugging is required, switch the jumper cap to the SWD end below.

1.1.2. The START development board

The START development board is composed of a baseboard and a module equipped with GD32W51x WiFi chip. There are four types of GD32W51x WiFi chip, either internal SIP FLASH or external FLASH, and either 36 PIN package or 56 PIN package.

Figure 1-2. The picture of the START development board
For developers, the main focus may be on the following sections of the development board, which are indicated in the picture above.

- Boot mode (Boot PIN)
- Power supply port (Power supply)
- Querying log (LOG UART)
- Debugger interface (DAPLINK or JLINK)
- Reboot (Reset Button)

1.2. Boot mode

The GD32W51x chip can be booted from ROM, FLASH or SRAM.

The level selection of the BOOT0 and BOOT1 pins of the development board determines the boot mode, as shown in Table 1-1. Boot mode. For more information about the boot mode, see the “GD32W51x User Manual”.

<table>
<thead>
<tr>
<th>BOOT0 pin is high</th>
<th>BOOT1 pin is high</th>
<th>SRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOT0 pin is low</td>
<td>BOOT1 pin is low</td>
<td>ROM (Legacy Bootloader)</td>
</tr>
</tbody>
</table>

1.3. Debugger interface

For EVAL development board, GD-LINK or JLINK can be used as the debugging interface. The development board comes with the GD-LINK debugger, which can be used by simply connecting the Mini USB power port. The JLINK SWD mode is supported, leading out four pins +3V3, IO, CK and GND respectively, which correspond to JLINK emulator pins VCC, SWDIO, SWCLK and GND, as shown in Figure 1-3. Pins of J-LINK.

For START development board, DAPLINK or JLINK can be used as the debugging interface. The development board comes with the DAPLINK debugger (GD32F303), which can be used by simply connecting the power port Micro USB. The DAP chip is also integrated with UART, so only one USB cable is needed to complete power supply, debugging and log viewing. JLINK SWD leads out four pins, which are AGND, JCLK, JTWS and W3V3. Connect the four pins with GND, SWCLK, SWDIO and VCC (As shown in Figure 1-3. Pins of J-LINK) corresponding to the J-LINK emulator by using Dupont wires, and then Burning and debugging code can be done via J-Link SWD mode. Connect pins JCLK and JTWS with the lower two pins respectively through jumper caps to download and debug code through DAPLINK. Figure 1-2. The picture of the START development board shows the debugging through DAPLINK.
1.4. View log

Use the MiniUSB cable to connect the EVAL development board or the MicroUSB cable to connect the START development board. Then start the serial port tool on the PC and configure parameters according to the Figure 1-4. The serial port configuration and connect the development board. After the connection is successful, logs can be printed through the serial port.

Figure 1-4. The serial port configuration
2. **Build development environment**

Before compiling and burning the firmware, the development environment needs to be set up.

At present, there are three commonly used development tools, namely KEIL, IAR and GCC. The SDK for GD32W51x already supports all three development tools. Each of the three tools has its own strengths and can be selected according to specific requirements.

2.1. **Keil MDK5 installation**

- **Download**
  - Official website: [https://www2.keil.com/mdk5](https://www2.keil.com/mdk5), Please select MDK525 and later.
  - If forward compatibility with MDK4 is required, please download MDKCM5xx.EXE from the following website: [https://www2.keil.com/mdk5/legacy](https://www2.keil.com/mdk5/legacy).

- **Installation**
  - The installation option can be selected as default, KEIL MDK5 which is installed under the “C:/Keil_v5” directory.
  - GD32W51x PACK
  - The PACK file is located in the GD32W51x_Addon / KEIL folder in the SDK, that is, GigaDevice.GD32W51x_DFP_1.x.x.pack.
  - Double-click the PACK file to run it, click the button "Next", as shown in Figure 2-1. **Install the KEIL GD32W51x PACK**. After installation, the PACK file is installed under “C:/Keil_v5/ARM/PACK/GigaDevice”.

![Figure 2-1. Install the KEIL GD32W51x PACK](image)
2.2. IAR Installation

- Download
  - Select version 8.32 or later to better support ARM CM33.

- Installation
  - Default installation is optional. IAR will be installed under "C: / Program Files (x86) / IAR Systems / Embedded Workbench 8.2".

- GD32W51x Addon
  - Addon is located in the GD32W51x_Addon / IAR folder in the SDK, that is, IAR_GD32W51x_ADDON_1.x.x.exe.
  - Right-click "Run as Administrator", select the path to the IAR installation path, and click "start" to complete the installation, as shown in Figure 2-2, Install the IAR GD32W51x Addon.

Figure 2-2. Install the IAR GD32W51x Addon

2.3. CMAKE+GCC installation

For the GCC environment, CMAKE and MAKE are selected, and use GCC Toolchain to compile.

- CMAKE installation
  - Download website: https://cmake.org/download/.
  - Select version 3.15 or later. cmake-3.20.3-windows-x86_64 is recommended.

- MAKE installation
  - make-3.81 is recommended.
Toolchain installation

Add to the system path
- After the installation is successful by using the default path, add the following path to the path of the system environment variables. If the installation path is changed, add the corresponding installation path to the system path.
  a) C:/Program Files (x86) / GnuWin32 / bin
  b) C:/Program Files (x86) / GNU Tools Arm Embedded / 9 2020-q2-update / bin
  c) C:/Program Files / CMake / bin

If JLINK is used as the debugger, the JLINK driver needs to be installed, refer to JLINK driver installation. If GD-LINK or DAPLINK is used, OpenOCD is used for downloading and debugging. OpenOCD does not need to be installed. The execution file is located in GD32W51x_Addon / openOCD / bin / openocd.exe.

2.3.1. JLINK driver installation

Developers can choose to use JLINK for development debugging, if not, skip this section

If the JLINK driver is not installed or its version is earlier, perform the following steps to download and install the JLINK driver.

Download JLINK driver
- Download JLink_Windows_V648b driver, as shown in Figure 2-3. JLINK driver download. Select Windows V6.48b and click “DOWNLOAD”. On the page that pops up, select accept the protocol and download it directly.

Figure 2-3. JLINK driver download

Install J-LINK driver
- Double-click JLink_Windows_V648b.exe to install the Jlink driver. Select the “Ok” option only on the screen shown in Figure 2-4. JLINK driver installation and the default option for the rest. Make sure JLink.exe is installed in C:/Program Files
Figure 2.4. JLINK driver installation

- JLINK flash burning configuration
  - To use JLINK to burn flash in the GCC development environment, go to the GD32W51x_Release/scripts/directory and run the “setup_jlink.bat” file by using rights of administrators. The FLASH burning algorithm file and JLinkDevices.xml will be replaced in the JLink installation directory “C:/Program Files (x86)/SEGGER/JLink/”.

3. Development instructions

Before the development, first understand the members of the SDK execution program group, how to correctly configure the SDK, and the final firmware generation location and composition.

SDK compilation and debugging can be done according to different development tools, such as KEIL, IAR or GCC, please select the corresponding chapter to understand.

3.1. SDK execution program group

The SDK finally generates two executables: MBL_NS (Main Bootloader) and WIFI_IOT (Some configuration files are called NSPE to be consistent with the name in the Trust Zone enabled mode). They all end up being burned to FLASH. After powering on, the program will start from the Reset_Handler of MBL_NS and then jump to the WIFI_IOT main program to run, as shown in Figure 3.1. Boot process.

Figure 3-1. Boot process
3.2. SDK configuration

3.2.1. Platform configuration

The configuration file is GD32W51x_RELEASE / config / platform_def.h, the main content is shown in Figure 3-2, Platform configuration.

Figure 3-2. Platform configuration

```
#define CONFIG_PLATFORM PLATFORM ASIC 32W51X

#ifndef CONFIG_PLATFORM
#error "CONFIG_PLATFORM must be defined!"
#endif

#define CONFIG_PLATFORM ASIC

#define PLATFORM_BOARD_32W515T_START 0
#define PLATFORM_BOARD_32W515P_EVAL 1

#define CONFIG_BOARD PLATFORM_BOARD_32W515P_EVAL

#define XIP_FLASH_SIP 0
#define XIP_FLASH_EXT 1
#define CONFIG_XIP_FLASH XIP_FLASH_SIP

#ifndef CONFIG_XIP_FLASH
#error "CONFIG_XIP_FLASH must be defined!"
#endif

#define QSPI_FLASH_1_LINE 0
#define QSPI_FLASH_2_LINES 1
#define QSPI_FLASH_4_LINES 2
#define QSPI_FLASH_MODE QSPI_FLASH_4_LINES

#define CRYSTAL_26M 0
#define CRYSTAL_40M 1
#define PLATFORM_CRYSTAL CRYSTAL_40M

#if defined(CONFIG_BOARD) && (CONFIG_BOARD == PLATFORM_BOARD_32W515P_EVAL)
#define LOG_UART_USART2
#else
#define LOG_UART_USART1
#endif

#ifndef CONFIG_PLATFORM_FPGA
#define CONFIG_PLATFORM_FPGA
#endif

#define RFAD SPI SPI0

#define TOTAL_SRAM_SIZE (448 * 1024)
```

- For the EVAL development board, please select:
  - #define CONFIG_BOARD PLATFORM_BOARD_32W515P_EVAL

- For the START development board, please select:
  - #define CONFIG_BOARD PLATFORM_BOARD_32W515T_START
- For the built-in SIP FLASH, please select:
  - #define CONFIG_XIP_FLASH XIP_FLASH_SIP

- For the external EXT FLASH, please select:
  - #define CONFIG_XIP_FLASH XIP_FLASH_EXT

- For crystal is 40 MHz, please select:
  - #define PLATFORMCRYSTAL CRYS_40M

- For crystal is 26 MHz, please select:
  - #define PLATFORMCRYSTAL CRYS_26M

### 3.2.2. SRAM layout

The configuration file is GD32W51x_RELEASE / config / config_gdm32_ntz.h. Modify the following macro definition values to plan the SRAM space occupied by the executable program segments MBL and NSPE. These values are offset addresses, and the base address is defined at the beginning of the file.

NSPE refers to the WIFI_IOT execution program segment, and Non-Secure Process Environment (NSPE) is the name of the WIFI_IOT execution program segment after the Trust Zone is enabled. As opposed to Secure Process Environment (PROT as seen in the configuration file), WIFI_IOT runs in a non-secure environment.

The line marked "! Keep unchanged!" cannot be modified, otherwise it will affect the running of the code MbedTLS in the ROM.

**Figure 3.3. SRAM layout**

```c
/* SRAM LAYOUT */
#define RES_MBL_DATA_START 0x200 /* !Keep unchanged! */
#define RES_NSPE_DATA_START 0x200 /* !For SIP flash! */
```

The SRAM space plan inside each executable segment can be viewed in Scatter File on the Linker page of the Project option. The macro definitions in the file can be found in "xxx_region.h", for example, "nspe_region.h".

### 3.2.3. FLASH layout

The configuration file is GD32W51x_RELEASE / config / config_gdm32_ntz.h. Modify the following macro definition values to plan the FLASH space occupied by the executable program segments MBL and NSPE. These values are offset addresses, and the base address is defined at the beginning of the file. NSPE is explained in **SRAM layout**.

The line marked "! Keep unchanged!" cannot be modified, otherwise it will affect the running of the project.
The FLASH space plan inside each executable segment can be viewed in Scatter File on the Linker page of the Project option. The macro definitions in the file can be found in "xxx_region.h", for example, "nspe_region.h".

3.2.4. Firmware version number

The configuration file is GD32W51x_RELEASE/config/config_gdm32_ntz.h. Modify the following macro definition values to specify the version number. However, the only version number that will affect future user upgrades is RE_NSPE_VERSION.

The MBL can only be upgraded locally, and the NSPE can be upgraded online. The SDK version is the same as RE_NSPE_VERSION, and the version number is displayed in the startup log through UART, for example, “SDK version: V1.0.0”.

3.2.5. APP configuration

The configuration file is “GD32W51x_RELEASE/NSPE/WIFI_IOT/app/app_cfg.h”. Some applications can be opened or not, for example: ATCMD, SSL example, Aliyun, FATFS and so on.

3.3. Firmware generation and download

The BIN and HEX files automatically generated by the compiled script are placed in the “GD32W51x_RELEASE/scripts/images/” directory. Image-* .bin is used for production or upgrade.

The "image-all.bin" file contains executable program segments MBL_NS and WIFI_IOT, which can be used for production and burned to blank FLASH.

The “image-ota.bin” file contains only WIFI_IOT (NSPE), which can be used to upgrade.
For download, in addition to the IDE tool download, for the START development board, can also be downloaded by the copy and paste method. When the development board is plugged into the computer via USB cable, the DAPLINK disk as shown in Figure 3-6. List of devices and drivers. Directly copy the” image-all.bin” file into the DAPLINK disk to complete the FLASH burning of GD32W51x chip. HEX file downloads are also supported.

Figure 3-6. List of devices and drivers

3.4. Example of correct log

After the firmware group (MBL_NS+WIFI_IOT) is downloaded successfully, open the serial port tool and press the “Reset” button on the development board. The startup information is shown in Figure 3-7. Project boot Information. If an exception occurs, refer to Q&A to try to resolve it.

Figure 3-7. Project boot Information
4. **KEIL project**

This chapter describes how to compile and debug the SDK in KEIL.

The project group consists of two projects: MBL_NS and WIFI_IOT. WIFI_IOT contains the WiFi protocol stack, peripheral drivers, applications, etc. MBL_NS is mainly responsible for selecting the correct firmware to run from two WIFI_IOT firmware (one for the current firmware and one for the upgraded firmware).

4.1. **Project group**

To enable KEIL to identify GD32W51x, install “GD32W51x_DFP_1.x.x.pack” in the “gigadevice.GD32w51x_addon/KEIL” directory before opening the project.

Find “MultiProject_NS.uvmpw” from the root directory of GD32W51x_RELEASE and double-click KEIL to open the workspace, as shown in Figure 4-1. KEIL project Group interface.

Figure 4-1. KEIL project Group interface
4.2. Project configuration

4.2.1. Device selection

The GD32W515 chip is available in a variety of types and can be seen on the KEIL Options->Device page once the PACK is installed. As shown in Figure 4-2, KEIL “Device” options page. The default type is GD32W515PIQ6, 2M SIP FLASH, and 448K SRAM.

If the chip is of another type, select it based on the actual type.

Figure 4-2. KEIL “Device” options page

4.2.2. Debugger configuration

For the EVAL development board, select on-board GD-LINK and for the START development board, select on-board DAPLINK. The Debugger on the Debug page is “CMSIS-DAP ARMv8-M Debugger”. For JLINK, select “J-LINK/J-TRACE Cortex”, as shown in Figure 4-3, KEIL “Debug” options page.
In the KEIL project, relevant Debugger settings have been configured by default. Note that the JLINK port must be selected as SW. Use JLINK Debugger, connect GD32W515 chip, the page as shown in Figure 4-3. KEIL “Debug” options page will appear.

If the Device is changed, the Debugger settings need to be reconfigured. After selecting the correct Debugger, the FLASH download algorithm needs to be selected, as shown in Figure 4-4. KEIL “JLINK Settings Debug” options page.
4-5. KEIL “Flash Download” options page. Note that the "Size" in "RAM for Algorithm" needs to be changed to 0x2000.

Figure 4-5. KEIL “Flash Download” options page

4.3. Compilation

If a project needs to be compiled, verify that the project is currently active project. If not, select the project to be compiled, right-click it, and click “Set as Active Project”, as shown in Figure 4-6. Current project switch. Then click the icon  to compile the project.
Batch compilation

Batch compilation is an option, as shown in Figure 4-7, Batch compilation. After clicking “Batch Build”, Project_MBL_NS and Project_WIFI_IOT are compiled in turn. Click on the “Batch Setup...” ,custom batch compilation is available, as shown in Figure 4-8, Batch compilation settings. The OS for Project_WIFI_IOT can be FreeRTOS or RT-Thread.

Figure 4-7. Batch compilation
User Command

The “User Command” is defined on each Project and automatically runs xxx_afterbuild.bat after compiling and linking successfully. This script mainly extracts from the output AXF file to generate the BIN format firmware. If it is nspe_afterbuild.bat, it will do one more thing, which is to concatenate mbl_ns.bin and nspe.bin to get image_all.bin.

4.4. Download

Click the icon 🔄 to download. Check whether the download is successful, as shown in Figure 4-9. KEIL download result.

Figure 4-9. KEIL download result

4.5. Debug

Initialization File

Before debugging, the “Initialization File (xxx.ini file) ” needs to be edited. For cross-project debugging, the AXF files of multiple projects need to be loaded in, see GD32W51x_RELEASE / MBL / project / KEIL / mbl_ns.ini, as shown in Figure 4-10. Initialization file for debugging.
Start debugging

After the above ready, click on the icon 📌, Or click "Debug" and then "Start / Stop Debug Session" to start single-step debugging, as shown in **Figure 4-11. KEIL debugging interface**.
5. **IAR project**

This chapter describes how to compile and debug the SDK in IAR.

The project group consists of two projects: MBL_NS and WIFI_IOT. WIFI_IOT contains the WiFi protocol stack, peripheral drivers, applications, etc. MBL_NS is mainly responsible for selecting the correct firmware to run from two WIFI_IOT firmware (one for the current firmware and one for the upgraded firmware).

5.1. **Project group**

Find MultiProject_NS.eww from the GD32W51x_RELEASE root directory and double-click it to open the IAR workspace, as shown in [Figure 5-1. IAR project group](#).

Please confirm that IAR_GD32W51x_ADDON_1.x.x.exe has been installed.

![Figure 5-1. IAR project group](#)

5.2. **Project configuration**

IAR Project configuration, mainly reflected in "Options..." dialog box, which is the IAR option mentioned below. This includes Device selection, output file selection, compile options, link options, download and debug options, and more.
The main things that may need to be adjusted are Device selection and debugger configuration.

5.2.1. Device selection

The GD32W515 chip is available in a variety of types and can be seen on the “IAR Options -> General Options -> Target” page once the Addon is installed, as shown in Figure 5.2. IAR “Target” options page. The default type is GD32W515xl, 2M SIP FLASH, and 448K SRAM.

If the chip is of another type, select it based on the actual type.

Figure 5.2. IAR “Target” options page

5.2.2. Debugger configuration

For the EVAL development board, select on-board GD-LINK and for the START development board, select on-board DAPLINK. The Driver on the Debugger Setup page is “CMSIS-DAP”.

For JLINK, select “J-LINK/J-TRACE Cortex”. As shown in Figure 5.3. IAR “Debugger” options page.
5.3. Compilation

To compile, just right-click on the corresponding Project and select "Make". To recompile all, select "Rebuild All", as shown in Figure 5-4. IAR compilation.
Batch compilation

Batch compilation needs to be set up first, click "Project" on the menu bar and select "Batch build..." from the drop-down menu and the “Batch Build” dialog box will pop up, as shown in Figure 5-5. IAR batch compilation settings. After clicking "New", the "Edit Batch Build" dialog box will pop up. Select all projects to the right box, fill in "Name" and click "OK". Go back to the “Batch Build” dialog box, select “All” and click “Make” to compile Project_MBL and Project_WIFI_IOT successively.
Build Actions

Each Project user can customize "Build Actions", as shown in Figure 5-6. IAR Build Actions. Currently, the "Pre-build command line" and "Post-build command line" are defined on each project, which are executed before and after compilation, respectively. Xxx_prebuild.bat" is used to complete ICF file precompilation and generate compilation timestamps. "Xxx_afterbuild.bat" is mainly extracted from the output AXF file to generate the BIN format firmware. If it is nspe_afterbuild.bat, it will do one more thing, which is to concatenate mbl_ns.bin and nspe.bin to get image_all.bin.
5.4. Download

Click "Project" and then "Download active application" to download. The correct burning Log displayed in the IAR Debug Log window is shown in Figure 5-7. IAR download result.

Figure 5-7. IAR download result

<table>
<thead>
<tr>
<th>Reset:</th>
<th>Half core after reset via DEMCR_VC_CORERESET.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset:</td>
<td>Reset device via ARPRG_SYSGENRESET.</td>
</tr>
<tr>
<td>Hardware reset with strategy 0 was performed.</td>
<td></td>
</tr>
<tr>
<td>Initial reset was performed.</td>
<td></td>
</tr>
<tr>
<td>952 bytes downloaded (12.42 Kbyte/sec).</td>
<td></td>
</tr>
<tr>
<td>Loaded debug: C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.2\arm\config\flashloader\GD\Flash\GD32W51x\S4RAM\448K.out.</td>
<td></td>
</tr>
<tr>
<td>Target reset.</td>
<td></td>
</tr>
</tbody>
</table>

5.5. Debug

- Debug

Click “Debug without Downloading” to start single-step debugging, as shown in Figure 5-8. IAR debugging button and Figure 5-9. IAR debugging interface.
Figure 5-8. IAR debugging button

![IAR debugging button](image)

Figure 5-9. IAR debugging interface

![IAR debugging interface](image)

- Debugging of multiple images

For cross-project debugging, on the “Debugger”->"Images" page, fill in the "Path" box
with the output file (*.axf or *.out) path of the next containing debugging information, and fill in 0 for “Offset”, and select “Debug info only”, as shown in **Figure 5-10. IAR multi-image debugging**.

**Figure 5-10. IAR multi-image debugging**
6. **GCC project**

This chapter describes how to compile and debug the SDK in GCC.

Currently, only JLINK debugging is supported, and the CMSIS-DAP debugging function will be added later.

6.1. **Project group**

Using cmake + gnu make tools to build the SDK, there are two main targets: WIFI_IOT and MBL_NS. WIFI_IOT contains the WiFi protocol stack, peripheral drivers, applications, etc. MBL_NS is mainly responsible for selecting the correct firmware to run from two WIFI_IOT firmware (one for the current firmware and one for the upgraded firmware).

6.2. **Compilation**

Start the Windows command window, do not use PowerShell. Go to “GD32W51x_RELEASE” directory and run the following command:

- Creates the cmake build directory.
  ```
  mkdir cmake_build
  ```
- Enter the cmake build directory.
  ```
  cd cmake_build
  ```
- Run the cmake command to generate the Makefile.
  ```
  cmake -G "Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE: PATH =. / scripts / cmake / toolchain.cmake
  ```
- Run the make command to compile.
  ```
  make -j
  ```

The preceding steps have been written into gcc_build_ns.bat. Double-click it to complete the first compilation.

- If only nspe needs to compile, go to the cmake_build directory and run the following command.
  ```
  make nspe -j
  ```
- If only mbl needs to compile, go to the cmake_build directory and run the following command.
  ```
  make mbl-ns -j
  ```
  - Download

Start the Windows command window, do not use PowerShell. Go to “GD32W51x_RELEASE” directory and run the following command:

- gcc_download_ns.bat JLINK
6.3. **Debug**

6.3.1. **Start the GDB server**

The debugging tool is GDB + JLink. Ensure that the Jlink connection is correct.

**Conditions preparation:**

- Create an environment variable HOME, the recommended value is the user directory, such as C:/Users/leo, see [Figure 6-1. Create the environment variable HOME](#) for details.

**Figure 6-1. Create the environment variable HOME**

- In the directory pointed to by the environment variable HOME, create the file named ".gdbinit" with the following contents: set auto-load safe-path /.

The steps to start debugging are as follows:

- Start the JLink Server and double-click JLinkGDBServer.exe in the installation directory.
Click “OK” after the configuration is complete.

- C:/Program Files (x86)/SEGGER/JLink/JLinkGDBServer.exe.

Figure 6-2. Start the JLINK GDB Server

6.3.2. Start the GDB Client for debugging

After starting GDB server, the GDB client is ready for debugging. In addition, start the windows command window, go to the directory “GD32W51x_RELEASE / NSPE / Project / WIFI_IOT / GCC”, enter the command gdb, as shown in Figure 6-3. GDB debugging window:

- arm-none-eabi-gdb output / bin / nspe.axf.
- Press Enter when prompted until the program stops at Reset_Handler.
At this point, program debugging can be done by using the gdb command. The following are common command references, as shown in **Figure 6.4. Common commands**. For details, refer to the official document: [https://www.gnu.org/software/gdb/documentation/](https://www.gnu.org/software/gdb/documentation/).

**Figure 6.4. Common commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bt</td>
<td>打印调用栈信息</td>
</tr>
<tr>
<td>i r</td>
<td>打印寄存器信息</td>
</tr>
<tr>
<td>c</td>
<td>(继续)执行代码到下一断点处</td>
</tr>
<tr>
<td>si</td>
<td>单步指令执行</td>
</tr>
<tr>
<td>mi</td>
<td>按指令执行到下一个断点</td>
</tr>
<tr>
<td>b</td>
<td>在原文件断点设置断点，例如 b 5</td>
</tr>
<tr>
<td>b function</td>
<td>在函数入口处设置断点，例如 b FuncName</td>
</tr>
<tr>
<td>b + address</td>
<td>在指定内存地址处设置断点，例如 b 0x00000a00</td>
</tr>
<tr>
<td>b</td>
<td>不带参数，表示在下一条指令设置断点，在调用函数前断点立即中断</td>
</tr>
<tr>
<td>d id</td>
<td>查看断点信息</td>
</tr>
<tr>
<td>display /Si $pc</td>
<td>五分之二从PC开始的5条指令</td>
</tr>
<tr>
<td>x /Si $pc</td>
<td>五分之二从PC开始的5条指令</td>
</tr>
<tr>
<td>x/4 $pc</td>
<td>查看从PC开始的4条指令</td>
</tr>
<tr>
<td>x /4n address</td>
<td>查看内存信息，以16进制32位打印 address 处的内存信息</td>
</tr>
<tr>
<td>4表示打印次数，x表示16进制，n表示双字</td>
<td></td>
</tr>
<tr>
<td>disassemble</td>
<td>查看汇编代码</td>
</tr>
<tr>
<td>disassemble /m</td>
<td>查看汇编代码和源码</td>
</tr>
<tr>
<td>disassemble /c</td>
<td>查看汇编机器码</td>
</tr>
<tr>
<td>l</td>
<td>查看当前代码</td>
</tr>
<tr>
<td>l function</td>
<td>查看函数源代码，例如 l CORE_Starting</td>
</tr>
<tr>
<td>l linenumbers</td>
<td>从当前文件开始打印基本源代码，例如 l 123</td>
</tr>
</tbody>
</table>
7. Q&A

7.1. DAPLINK disk identification

When the START development board is used, if the serial port cannot be identified as “mbedSerialPort (COMx)” and the operating system is Windows 7 or later, install the Mbed serial port driver.

- **Driver download**

- **Installation**
  - Double-click “mbedWinSerial_16466.exe”, and then click “Install”, as shown in Figure 7-1. Mbed serial port driver installation. After the installation is successful, click “Finish”.

![Figure 7-1. Mbed serial port driver installation](image)

- **Check**
  - After the installation is successful, “mbed Serial Port (COMx)” is displayed on the device manager, as shown in Figure 7-2. List of device manager serial ports. The "DAPLINK" disk can be seen in the list of devices and drivers, as shown in Figure 3-6. List of devices and drivers.

![Figure 7-2. List of device manager serial ports](image)
7.2. CM33 debugging by KEIL+JLINK

By default, MDK5.25 does not support debugging ARM CM33 with JLINK. When ARM CM33 needs to use JLINK debugging, if the option to select JLINK debugger cannot be found in the debugger selection column on the Debug page of KEIL tool engineering options, as shown in Figure 7-3, KEIL DEBUG options, the "C:/Keil_v5/TOOLS.INI" file needs to be modified.

Figure 7-3. KEIL DEBUG options

Add TDRVx corresponding to JLINK in line CPUDLL3 of [ARMADS] in TOOLS.INI file to increase support for JLINK. Using JLINK debugging development board needs to add "TDRV4", according to Figure 7-4, Add tools supported by KEIL, and add "TDRV4" to the corresponding position. After the modification, re-open the KEIL tool and select "J-LINK/J-TRACE Cortex" to debug ARM CM33.

Figure 7-4. Add tools supported by KEIL
7.3. **DAP chip firmware upgrade**

If the DAP chip on the START development board needs firmware upgrade, keep the “Reset” button on the development board pressed, and then use the USB cable to power on, the computer will detect the MAINTENANCE disk at this time, as shown in Figure 7-5, **MAINTENANCE disk**.

Copy gd32w51x_if_crc_qspi.bin (for chip GD32W51x EXT FLASH) or gd32w51x_if_crc_fmc.bin (for chip GD32W51x SIP FLASH) to the MAINTENANCE disk. Upon success, the new DAP firmware will run and the DAPLINK disk will appear.

![Figure 7-5. MAINTENANCE disk](image)

7.4. **No image error**

Print ERR: No image to boot (ret = -5).

**Cause**: The previous booting WIFI_IOT failed and MBL recorded that the IMAGE was running abnormally. If the other IMAGE was not burned or a booting exception occurred, it would print this message. That is to say, MBL considers that there is no legal IMAGE to jump to, and the boot fails.

**Solution**: MBL can be burned again, after burning the IMAGE state will be cleared.

7.5. **Code runs in SRAM**

If programs need to run faster to achieve higher performance, consider moving them to the SRAM.

- **Use KEIL**
  
  Open GD32W51x_RELEASE / NSPE / Project / WIFI_IOT / KEIL / Project.sct for editing, add the required files or functions to the Execution Region of RW_IRAM2, such as:
  - port.o (+RO)
    
    Put the entire port.c file into the SRAM to run.
  - tasks.o (.text.xTaskIncrementTick)
    
    Put the xTaskIncrementTick () function in tasks.c into the SRAM to run.

  If the function is not added successfully, check the “GD32W51x_RELEASE / nspe / Project / WIFI_IOT / KEIL / wifi_iot / Freertos / List / nspe.map” and find the correct
symbol name according to the function name, as shown in Figure 7-6. NSPE MAP file.

---

Use IAR
If some functions need to be run in SRAM, add SECTION_RAM_CODE to the end of the function, such as:
- BaseType_t xTaskIncrementTick (void) SECTION_RAM_CODE

Use GCC
Open “D32W51x_RELEASE / NSPE / Project / WIFI_IOT / GCC / nspe_gdm32_ns.ld” and find the line ".code_to_sRAM: ". The brace contains code that runs in SRAM. If new content needs to be added, it can be added at the end. Refer to existing documents in the format, such as:
- KEEP (*port.o* (.text* .rodata*))
  Put the entire port.c file into the SRAM to run.
- KEEP (*tasks.o* (.text.xTaskIncrementTick))
  Put the xTaskIncrementTick () function in tasks.c into the SRAM to run.
8. Revision history

Table 8-1. Revision history

<table>
<thead>
<tr>
<th>Revision</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Initial Release</td>
<td>Mar.25, 2022</td>
</tr>
</tbody>
</table>
Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the "Product"), is owned by the Company under the intellectual property laws and treaties of the People’s Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as it’s suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as it’s suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.

© 2022 GigaDevice – All rights reserved